To marshal and use resources best in solving problems of automation or mechanization--at least those that include an ADP or information-processing part--requires a close look at the state of the new computer science and at the roles its disciples customarily play best (including that of rivals), as well as what customers and management should expect as results. People who know ADP systems and their applications as well as most contractors do may be found throughout the Agency. They can help our own ADP people. But in the best case, where an obviously important problem is well-defined and attacked by cooperative efforts, only management direction, restraint, and review will ensure that the defined problem is solved with economy of design and operation. That management direction and review comes best from one(s) attached to the front office, serving its broad interests directly--possibly as its coordinator of systems development.

Despite widespread interest and activity, it's no secret that the computer sciences are still in the early stages of development. Principles of systems analysis, or the application of operations research methods to make complex mechanisms work with economy of design and operation, are often ignored in practice--at least the "economy" part, by systems analysts. Programming is less a science than a set of procedures of varying efficiencies, and programming rarely uses more than a fraction of the computer power. Vendor

software is notoriously inefficient, and requires maintenance as it interacts--often badly--with applications software. Contractor software frequently won't run in a production environment without considerable modification by our own programmers. ADP people can rarely provide efficient results from the start of their system in a production environment, and must continue to eliminate errors and tune up their system after it starts. To start writing software with their current languages, programmers require problems defined with the utmost specificity. So some detailed description of the problem, as well as any mathmetical formulas required in its solution, and system analysis must attempt to bridge the gap between the user's normally broad concept of his need and the programmer's detailed translation of it.

This current state of affairs argues for developing and using our own Agency expertise if at all possible, because management can direct a more continuing and thorough effort, including redesign or reprogramming of systems, than they could afford with a contractual effort. It also argues that programmers and programming management should concentrate on efficient use of the computers by tuning up their operating or executive systems, and by writing only those programs that they alone can write efficiently*--letting users write programs that won't hurt efficiency. (Indeed, future ADP units may spend almost all of their time installing and maintaining executive systems and general file management systems.**)

It argues for users' writing what programs they can in an

*Programmers alone can write programs required for large file management and data manipulation, like the IIS programs, but many analysts can write fairly efficient FORTRAN programs for small calculations.

**General File Management Systems enable users to define, create, & maintain files, compute, manipulate data with no/minimum programmer help.

"open-shop" environment (with management constraints for efficiency), to conserve scarce programmer manpower and the time it would take to translate user needs for programmers. And it argues for a systems-development effort to ensure that customers get a broad understanding and thorough analysis of their problem.

To develop an efficient ADP-systems response to a customer need requires the efforts of experienced people with different skills, cooperating closely under management direction and review. No one "expert" is likely to possess all the required skills in the required measure to define the user's problem validly with details, to maintain a breadth of interest and knowledge about ADP, and to program with competence. The effort normally requires senior user analysts, to define what could best augment their own capabilities and methods, and what that augmentation might be worth in terms of anticipated products. Systems analysts should provide the breadth of ADP interest and knowledge, know what other disciplines can best provide, and keep up with current technology. Programming experts should know not only what is practical in their ADP environment, but what is likely to be economical and efficient in the near future. People with these individual skills are both costly and scarce, so their individual skills should be focused on that part of the problem they can do best and not diffused across the whole problem.

Users, systems analysts, and programmers on the team should learn as much as practical about each other's fields, not so they can concentrate on the others' job and lose some facility with their own, but so they can communicate more readily with the others. Ready communication can save documentation and writing in the initial problem-exploration period, can also help when they document their system.

Getting these people to solve the defined problems economically can be difficult when they work together. Let them play the role of competitors or rivals, and their individual skills will be partially wasted while they struggle to do the whole job for which they're probably not by training or temperament prepared. Moreover, two or more separate solutions--or none--may result. And these solutions will have to be tested, modified, and retested before the customer or management can ever decide which, if any, is best.

If a customer can ask different people or units for help, those people may well compete to solve his whole problem, if not for all of his business. But by themselves, they are unlikely to solve his problem the "best" way. And the customer should not have to ponder over which unit to approach and why, for his particular problem. He wants one, helpful, answer from one point. Actually, cooperation may replace competition among the problem-solvers if the customer has only one point, authorized by management, to approach for systems help (which obviously does not prevent

him from discussing his problem with any person informally). That one point for customer contact should remain outside units that are potential competitors if it is to remain free of prejudice.

To ensure that a single systems-development team works cooperatively on valid user needs requires periodic, if not continuous direction, restraint, and review by a representative of top management; because units with ADP interests naturally want to do the whole job and have it accepted without question. The management man can coordinate individual or unit efforts, advise on constraints and intentions, and arrange for review and appraisal of their results with subsequent acceptance or redirection of effort. Management will want periodic reviews of the developmental work, to ensure that it does not get off the track, at such recommended milestones as these:

- definition of the problem and its size (all parties should agree or record their disagreement and reasons);
- assignment of priority and work force to the problem, or its dismissal;
- definition of the current system, its cost, and its failings (the reasons for considering another solution);
- definition of requirements for a new system;
- design and cost-estimates (rough) of new system alternatives;
- recommendation and approval or rejection, with subsequent redirection of any continued system-development effort.

To provide these services, management will want representation attached to itself.